REMARKS

Claims 1-18 remain pending in the patent application.

The Indefiniteness Rejection

In response to the points raised on page 2 of the Office Action, the claims are amended, as follows:

Claim 1 is amended to recite a mobile terminal featuring an operating system having a file system with one or more modules configured for detecting a filename with an illegal character, and for encoding the filename by replacing the illegal character with a specific code character having information coded therein about the illegal character itself. The terms "for" and "wherein" have been eliminated from the claim. Claim 15 is similarly amended.

Claim 6 is amended to eliminate the term "same".

Claim 16 is amended to recite that the decoder forms part of a file system.

Claim 17 is amended to recite a method featuring steps of detecting a filename with an illegal character in a file system of an operating system in a mobile terminal; and encoding the filename by replacing the illegal character with a specific code character having information coded therein about the illegal character itself so as to form an encoded filename. The results of the claimed method is an encoded filename according to the present invention.

The Statutory Subject Matter Rejection

Consistent with that stated above, claim 17 is amended to recite a method featuring steps of detecting a filename with an illegal character in a file system of an operating system in a mobile terminal; and encoding the filename by replacing the illegal character with a specific code character having information coded therein about the illegal character itself so as to form an encoded filename. The useful results of the claimed method is an encoded filename according to the present invention.

The Claimed Invention

Claim 1 recites a mobile terminal featuring an operating system having a file system with one or more modules configured for detecting a filename with an illegal character, and for encoding the filename by replacing the illegal character with a specific code character having information coded therein about the illegal character itself. Claims 15-17 contain similar limitations.

Claim 19 is added to recite a mobile terminal according to claim 1, wherein the specific code character is an n-bit coded character having a position field with one group of the n bits containing encoded information about the position of the illegal character in the filename, and a character field with another group of the n bits containing encoded information about the illegal character itself in the filename.

Claim 20 is added to recite a mobile terminal according to claim 19, wherein the n-bit coded character includes an identifier field with a further group of the n bits containing encoded information about the identity of the specific code character from other characters in a set of standard characters.

The claimed invention provides a simple symmetrical encoding for filenames to replace each illegal character with a respective specific code character.

Typically, filenames use 16 bit Unicode characters so there is 16 bits to encode one illegal character. According to the present invention, the encoder represents the position of the illegal character in the filename with 8 bits (2^8 = 256) and maps the illegal character itself using four bits that represent the Unicode character, as described in the patent application on page 3, lines 10-16.

In operation, the encoder identifies the specific code character from other characters by using the four most significant bits (MSB) as an illegal character indicator, e.g. using the Unicode designation "1110 = E". In the Unicode standard, code areas from E000 to F8FF are reserved for private or internal business use, and available for use as such an indicator.

In one embodiment, the specific code character can be placed at the end of the filename, before the commonly used filename extension. For example, this would make an illegal filename "ale?x.jpg" look like "alex .jpg", where the blank space represents the coded character.

The solution makes it possible to receive files, containing illegal characters, from an external source connected to the mobile terminal through WAP, UNIX Server or McIntosh and store the files into the file system without corrupting the filenames. The solution also makes it possible to decode the filenames back to the original format, recreating the illegal characters.

Moreover, the solution also makes it possible for the mobile terminal to be used as a mass storage device for popular Apple McIntosh operating system even though the file system in mobile terminal does not directly support the MacOS operating system.

In effect, the claimed invention relates to and involves an encoding modification technique where 16 bits (or 32/48/64/128 bits etc), which normally are used to store character mapping information, could be utilized in a different way as shown in Figure 2 of the patent application. Since there is only a restricted set of illegal characters, only four bits are needed to mark them (i.e. the claimed invention uses a customized character mapping) as illustrated in Figure 2a and 2b. In the example shown in Figures 2a, 2b, the four most significant bits (MSBs) are used as "illegal character indicator" when set in a given way e.g. ´1110´ and thus a filesystem knows that all characters within a filename that have "illegal character indicator" MSBs are treated in a special way (i.e. not as normal Unicode characters but as characters informing about illegal characters that have been removed from the filename).

Further, all those characters with "illegal character indicator" are shown as blank spaces by the filesystem (by contrast, in standard Unicode a blank space is 0x0020 but in the claimed invention those illegal characters coded as 0xE... are also shown as blank spaces). Consequently, filenames like "SunEarth .txt", "Sun>Earth.txt", "Sun?Earth.txt", "Sun:Earth.txt" all would look the same "SunEarth .txt" (if multiple illegal charaters then there is more blank spaces at the end of the filename, e.g. "Sun<>Earth?.txt" is shown as "SunEarth .txt"). So a user doesn't see any difference between those filenames and hence there may be several files with

similar looking names in the same directory (as opposed to Douceur's patent) but in bitlevel there are differences. In comparison, it is respectfully submitted that preventing duplicate filenames is one of the main advantages of Douceur's patent, so there is no teaching, suggestion or motivation to interpret Douceur's patent in a way the reasoning in the outstanding Office Action is suggesting.

As an example, the illegal character '>' in "Sun>Earth.txt" would be coded as '1110011100000011' (0xE703) and shown as a blank space at the end of the filename since 7 is used for '>' (see Figure 2b), and 3 is the position of the character since the numbering starts from 0 (i.e. fourth character in the filename has position 3, where with 8 bits one can locate illegal characters within filenames having length of 256 characters). If there were more illegal characters those would be coded with a similar syntax and placed at the end of filename in the order of appearance.

In the case of the filename "ale?x.jpg" which would be shown as "alex .jpg" in a filesystem not allowing '?' in filenames, those allowed characters 'a', 'l', 'e' and 'x' are coded in a standard way whereas '?' is removed and a special character having the "illegal character indicator" (e.g. 0xE) and shown as a blank space is coded as 0xE403 to indicate that it should be decoded as a '?' (see conversion table of Figure 2b) and placed as fourth character from the beginning of the filename (index 3) when transferred to a filesystem allowing such characters. In contrast, it is respectfully submitted that Douceur does not teach or suggest to use character bits in a similar way.

Moreover, the aforementioned discussion concentrates on a scheme where illegal characters are shown as blank spaces at the end of a filename. However, the scope of

the invention is not intended to be limited to the same. For example, one may replace an illegal character with a specific code character that has 'illegal character indicator' bits (4 MSBs) set in a given way (e.g. 1110). Thus, in such a situation "ale?x.jpg" would be shown as "ale x.jpg" and the character shown as blank space would be coded as 0xE403 in view of Figure 2.

The Prior Art Rejection

The main independent claims are rejected as being anticipated by Perrin et al. (US 2004/0088153).

However, it is respectfully submitted that Perrin et al. does not teach or suggest a file name encoding/decoding technique featuring replacing an existing character with a specific code character having information coded therein about the illegal character itself, as claimed herein, especially when the claimed limitation "character having information coded therein about the illegal character itself" is interpreted consistent with that shown and described in the patent application.

In contrast, Perrin et al. converts a requested file name into a valid Win32 file name (e.g., by replacing invalid Win32 characters with valid Win32 characters), and converts the file name back into the original file name, as described in section 45 of Perrin et al. In operation, Perrin has an emulator keep track of those replaced characters (the emulator has its own rules for character replacements, e.g. a conversion table). However, Perrin et al. does not teach or suggest making this replacement /conversion by replacing an existing character with a specific code character having information coded therein about the illegal character itself, as claimed.

Moreover, dependent claim 2 recites the subject matter of claim 1, wherein the specific code character includes information about the position of the illegal character in the filename. It is respectfully that in contrast to the claimed invention in Perrin the emulator keeps track of the character position. This character position information is not embedded in the specific code character itself.

Moreover still, dependent claim 8 recites the subject matter of claim 1, wherein the specific code character is placed in a predefined location in the filename. It is respectfully that in contrast to the claimed invention in Perrin the emulator keeps track of the character location. This character location information is not embedded in the specific code character itself.

Further, Perrin also does not teach or suggest that its replacement character is an n-bit coded character having a position field with one group of the n bits containing encoded information about the position of the illegal character in the filename, and a character field with another group of the n bits containing encoded information about the illegal character itself in the filename, as recited in newly added claim 19. In other words, Perrin's replacement character does not contain fields having coded information about the position and character of the illegal character itself.

Further still, Perrin does not teach or suggest that its replacement character is an n-bit coded character having an identifier field with a further group of the n bits containing encoded information about the identity of the specific code character from other characters in a set of standard characters, as recited in the newly added claim 20. In other words, Perrin's replacement character does not contain a field having coded

information about the identity of the illegal character itself from other characters in a set of standard characters.

With respect to other prior art, Underwood talks about replacing invalid characters with correct ones using a conversion table (Fig.9A), there is no hint or suggestion about "character having information coded therein about the illegal character itself." Underwood merely uses a conversion table.

Butterfield's prior art is about creating valid filename's by replacing illegal characters - it doesn't provide any hint or suggestion about "character having information coded therein about the illegal character itself".

In view of this, it is respectfully submitted that the reasoning clearly fails to show prior art that suggests/motivates/teaches a "character having information coded therein about the illegal character itself". All the cited prior art discloses are various approaches where invalid characters are replaced with valid ones using various conversion table definitions (in Perrin emulator does this - however, it is not explicitly mentioned; in Underwood there is that conversion table of Fig. 9A; in Butterfield invalid characters are just replaced with valid ones as there is no need to store information about invalid character).

Moreover, a means-plus-function apparatus claim is added as claim 18.

Remaining Dependent Claims

The remaining dependent claims depend directly or indirectly from one or more of the aforementioned independent claims, contain all the limitations thereof, and are deemed patentable for all the reasons set forth above.

The Amendment to the Specification

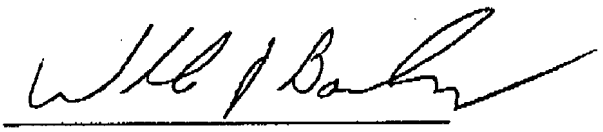
The paragraph bridging pages 1-2 is amended to correct a type, i.e. in line 1 thereof "know" is changed to --known--.

Conclusion

The commissioner is hereby authorized to charge any fees in order to submit this amendment to deposit account no. 23-0442.

For all these reasons, reconsideration and early allowance is respectfully requested.

Respectfully submitted,



William J. Barber
Attorney for the Applicant
Registration No. 32,720

10 September 2007
WARE, FRESSOLA, VAN DER SLUYS
& ADOLPHSON LLP
Customer No. 004955
Bradford Green, Building Five
755 Main Street, P.O. Box 224
Monroe, CT 06468
(203) 261-1234